CS2043 - Unix Tools & Scripting

Cornell University, Spring 2014[1]

Instructor: Bruno Abrahao

January 27, 2014

[1] Slides evolved from previous versions by Hussam Abu-Libdeh and David Slater

Recap - Commands

- `ls` - **lis**t directory contents
- `cd` - **c**hange **d**irectory
- `pwd` - **p**rint **w**orking **d**irectory
- `rm` - **rem**ove file
- `rmdir` **rem**ove **dir**ectory
- `cp` - **cop**y file
- `mv` - **mov**e file

Recap - Permissions

-rwxrwxrwx

- User's Permissions
- Group's Permissions
- Other's permissions

R = Read, W = Write, X = Execute

Directory Permissions begin with a `d` instead of a `-`

Recap - Changing Permissions - Convenience

Think of r, w, and x as binary variables:

- 1 ON
- 0 OFF

$$r \times 2^2 + w \times 2^1 + x \times 2^0$$

Examples

- `chmod 755` : rwxr-xr-x
- `chmod 600` : rw-------
- `chmod 777` : rwxrwxrwx

Default Permission

This command affects the permission of all files you create (during this session).

Umask

`umask mode`

- Removes mode from the file's permission.

Umask is the last-stage filter that strips away permissions as a file or directory is created where the bit is set to "1"

Examples

- `umask 077` : full access to user, no access to everyone else
- `umask g+w` : alternative notation: enables the group to have write permission
- `umask -S` : displays mask in simbolic notation

Recap - File Ownership

- Each file is assigned to a single user and a single group (usually written `user:group`).
- Needs root privilege to change file ownership.
- To see what groups you belong to type `groups` or `id`.

Changing Ownership

If you want to change the group of a file you have ownership of use the following

Change Group

```
chgrp group <target>
```

- Changes the group ownership of file `<target>`

If you have root access and you want to change who owns a file you use

Change Ownership

```
chown user:group <target>
```

- changes ownership of file `<target>`
- group is optional
- use the flag "-R" to do a recursive change to a directory and the files within

Recursion

Most commands (for which it makes sense) have a recursive option. This is used to act on every file in every subdirectory of the target

- Usually **-r** or **-R** option (check manpage)

Example:

```
chmod -R o-w ~/Documents/
```

- removes write privileges for other uses for every file and every directory in ~/Documents/

Link Files

Just like in windows, we can create links to files and directories. There are two types of links, hard links and symbolic links.

Link Creation

`ln [options] <target file> [link_name]`

- Creates a link to `<target file>` at `[link_name]`, defaulting to the current directory
- The link points to the same file on the system i.e. the link is indistinguishable from the original file
- In other words both the original file and the link both point to the same underlying object

Symbolic Links

Symbolic Link

```
ln -s <target_file> [link_name]
```

- Creates a symbolic link to the target file or directory
- The link file contains a string that is the pathname of the original file or directory
- In other words the symbolic link points to the other file

Types of files

There are two main types of files. The first is plain text files.

Text Files

Plain text files are written in a human-readable format. They are frequently used for

- Documentation
- Application settings
- Source code
- Logs
- Anything someone might want to read via a terminal

- Like something you would create in notepad
- Editable using many existing editors

Binary Files

Binaries

Binary files are written in machine code.

- Not human readable (at least without using hex editors)
- Commonly used for executables, libraries, media files, zips, pdfs, etc
- To create need some sort of binary-outputting program

What kind of file are you?

Displays the file type.

File

```
file <filename>
```

Let's explore some files on my system...

Reading Files

Often we only want to see what is in a file without opening it for editing.

Print a file to the screen

```
cat <filename>
```

- Prints the contents of the file to the terminal window

```
cat <filename1> <filename2>
```

- Prints the first file then the second which is what it is really for

More

```
more <filename>
```

- allows you to scroll through the file 1 page at a time

Less

```
less <filename>
```

- Lets you scroll up and down by pages or lines

Beginning and End

Sometimes you only want to see the beginning of a file (maybe read a header) or the end of a file (see the last few lines of a log).

Head and Tail

```
head -[numlines] <filename>
tail -[numlines] <filename>
```

- Prints the first/last numlines of the file
- Default is 10 lines

Example

```
tail /var/log/Xorg.0.log
```

Prints the last ten lines of the log file.

Printing to the terminal

We have already seen a variety of ways to print text to the screen. If we just want to print a certain string, we use

Echo echo... echo...

```
echo <text_string>
```

- Prints the input string to the standard output (the terminal)

-
  ```
  echo This is a string
  echo 'This is a string'
  echo "This is a string"
  ```
  all print the same thing
- We will see why we talk about these three cases later

Editing plain text

The shell is designed to allow the user to interact in powerful ways with plain text files. Before we can get to the fun stuff — do you want the blue or the red pill?

Dealing with plain text - Novice (blue pill)

Nano

```
nano filename
```

- Opens filename for editing
- In terminal editor
- Shortcuts for saving, exiting all begin with CTRL.
- This editor will do fine for everything we do in this course.

Turning Pro!

Vim = Awesome!

- Vim is a powerful lightweight text editor.
- The name “Vim” is an acronym for “Vi IMproved”
  - `vi` is an older text editor
- Ports of Vim are available for all systems
  - including Microsoft Windows

Vim allows you to perform text editing tasks much faster than most other text editors!

- Though it does have a learning curve

A modal text editor

- One of the reasons that `Vim` allows you to performs tasks quickly is because it works in modes.
- Without modes, users would have to either use command menus (with a mouse or keyboard), or use complex/long command shortcut keys involving the control key (`ctrl`) or the alt key (`alt`).
- `Vim` uses modes to speed up editing by not relying on command keys or menus.

You can do all of your editing by just using the keyboard which is super fast!!

The 3 main modes of Vim

- **Editing (normal) mode:**
  - Launching pad to issue commands or go into other modes
  - Allows you to view the text but not edit it
  - Vim starts in normal mode
  - You can jump to normal mode by pressing the `Escape` (`Esc`) key on your keyboard
- **Visual mode:**
  - Used to highlight text and perform operations on selected text
  - You get to visual mode from normal mode by pressing the `v` key on your keyboard
- **Insert mode:**
  - Used to type text into the buffer (file)
  - This probably what you're used to from your text editor
  - You get to the insert mode by pressing the `i` key on your keyboard

Moving around

Fast

You can use your mouse to move around in Vim (assuming a graphical interface as in `gVim`

Faster

However it is much faster to just use your keyboard, and for that you can just use the arrow keys to move up/down/left/right.

Fastest

You can even be more efficient by not leaving the main area of the keyboard and using “h” to go left, “j” to go down, “k” to go up, and “l” to go right.

To start off, I recommend you just use the arrow keys.

Basic commands

I can't possibly teach you about all the power of Vim in a few minutes, however here are a few commands to help you get started.

Getting help

```
:help
```

Entering normal mode

```
<Esc>
```

Entering insert mode (from normal)

```
<i>
```

Entering visual mode (From normal)

```
<v>
```

Basic commands

Save text to filename.txt

```
:w filename.txt
```

Exit

```
:q
```

Quit without saving

```
:q!
```

Open another file

```
:e [filename]
```

Getting started

To get started, launch Vim and go through the built in help.

```
:help
```

You can search for help on a specific topic as well.

```
:help [topic to search for]
```

Later in the course...

More to come:

- Useful everyday commands
- Piping, input/output redirection
- `ssh/sftp/scp`, `wget/curl`, `screen`
- And Much more!